

```

function tchains = runmcmc(pchains = [])
    % Driver code for MCMC analysis
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global Km
    global SVK
    global gg
    global preds

    numParms = 3
    numChains = 1
    numIts = 10000
    funcNames = ["mcInit", "mcEvalLikelihoods", "mcEvalPriors",
    "mcSamplePriors", "mcEvalProposal", "mcSampleProposal"]
    updateMode = 4
    chains = mcmc(numParms, numIts, numChains, updateMode, funcNames,
    pchains);
    save @format=ascii @file=mcmc_results.dat chains
    tchains = chains([1:1:10000],:);
end

function mcInit()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global Km
    global SVK
    global gg
    global preds
    global OpMcmcPriorBounds
    OpMcmcPriorBounds = [...
    0.01, 10
    -7, 2
    -10, 5
    ];
    global OpMcmcAdaptive
    OpMcmcAdaptive = 1;
    global OpMcmcDelayedRejection
    OpMcmcDelayedRejection = 0;
    global OpMcmcAdaptPeriod
    OpMcmcAdaptPeriod = 30;
    global OpMcmcAdaptCovarScale
    OpMcmcAdaptCovarScale = 1;

```

```

global OpMcmcLoggingPeriod
OpMcmcLoggingPeriod = 50;
global OpMcmcAdaptLowerThresh
OpMcmcAdaptLowerThresh = 0.25;
global OpMcmcAdaptUpperThresh
OpMcmcAdaptUpperThresh = 0.45;
global OpMcmcAdaptLowerThreshDR
OpMcmcAdaptLowerThreshDR = 0.45;
global OpMcmcAdaptUpperThreshDR
OpMcmcAdaptUpperThreshDR = 0.65;
global OpMcmcSigmaDecreaseFact
OpMcmcSigmaDecreaseFact = 0.9;
global OpMcmcSigmaIncreaseFact
OpMcmcSigmaIncreaseFact = 1.1;
global OpMcmcDRSigmaReduceFact
OpMcmcDRSigmaReduceFact = 0.2;
global OpMcmcDRSigmaReduceFactAM
OpMcmcDRSigmaReduceFactAM = 0.1;
global OpMcmcAdaptLowerThreshAM
OpMcmcAdaptLowerThreshAM = 0.15;
global OpMcmcAdaptUpperThreshAM
OpMcmcAdaptUpperThreshAM = 0.3;
global OpMcmcCovarScaleDecreaseFact
OpMcmcCovarScaleDecreaseFact = 20;
global OpMcmcCovarScaleIncreaseFact
OpMcmcCovarScaleIncreaseFact = 20;
global OpDemcSnookerFraction
OpDemcSnookerFraction = 0.1;
global OpDemcThinningFactor
OpDemcThinningFactor = 10;
global OpDemcB
OpDemcB = 0.0001;
end

function samp = mcSampleProposal (prevsamp)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global Km
    global SVK
    global gg
    global preds
    samp = [];
    % This function is a stub...
    % Code for a user-defined proposal function can be inserted here.
end

function val = mcEvalProposal(samp, prevsamp)
    global data

```

```

global firstT
global lastT
global firstD
global lastD
global CCC
global LI
global Vmax
global Km
global SVK
global gg
global preds
val = 0;
% This function is a stub...
% Code for a user-defined proposal function can be inserted here.
end

function mcDumpSamples()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global Km
    global SVK
    global gg
    global preds
    LI
    Vmax
    Km
end

function names = mcSampNames()
    names = "LI";
    names = [names, "Vmax"];
    names = [names, "Km"];
    names
end

function parms = mcPackSamples()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global Km
    global SVK
    global gg
    global preds

```

```

parms = [];
parms = [parms LI];
parms = [parms Vmax];
parms = [parms Km];
end

function mcUnpackSamples(parms)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global Km
    global SVK
    global gg
    global preds
    idx = 1;
    LI = parms(idx); idx = idx + 1;
    Vmax = parms(idx); idx = idx + 1;
    Km = parms(idx); idx = idx + 1;
end

function parms = mcSamplePriors()
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global Km
    global SVK
    global gg
    global preds
    LI = normrnd(1, 1, 0.01, 10);
    Vmax = unifrnd(-7, 2);
    Km = unifrnd(-10, 5);
    parms = mcPackSamples();
end

function val = mcEvalPriors(parms)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global Km

```

```

global sVK
global gg
global preds
mcUnpackSamples(parms);
val = 0.0;
val = val + normlpdf(LI, 1, 1, 0.01, 10);
val = val + uniflpdf(Vmax, -7, 2);
val = val + uniflpdf(Km, -10, 5);
end

function val = mcEvalLikelihoods(parms)
    global data
    global firstT
    global lastT
    global firstD
    global lastD
    global CCC
    global LI
    global Vmax
    global Km
    global sVK
    global gg
    global preds
    mcUnpackSamples(parms);
    val = 0.0;
    sVK = 0;
    gg = 1;
    for i = firstD(gg) : lastD(gg)
        preds = getpreds(Vmax, Km, sVK, CCC(i), gg);
        for j = firstT(gg) : lastT(gg)
            if(~isnan(data(j, i)))
                val = val + normlpdf(data(j, i), preds(j), LI);
            end
        end
    end
end

```